Amendments to the Specification:

Please replace paragraph **[0001]** with the following rewritten paragraph **[0001]**:

[0001] U.S. Patent Application Serial No. ~~_____~~10/672,698, filed September 25, 2003 in the name of inventor Eduard K. de Jong, entitled "Application Program Obfuscation", ~~Attorney Docket No. SUN-P7008,~~ commonly assigned herewith.

Please replace paragraph **[0002]** with the following rewritten paragraph **[0002]**:

[0002] U.S. Patent Application Serial No. ~~_____~~10/672,700, filed September 25, 2003 in the name of inventor Eduard K. de Jong, entitled "Permutation of Opcode Values for Application Program Obfuscation", ~~Attorney Docket No. SUN-040023,~~ commonly assigned herewith.

Please replace paragraph **[0003]** with the following rewritten paragraph **[0003]**:

[0003] U.S. Patent Application Serial No. ~~_____~~10/672,836, filed September 25, 2003 in the name of inventor Eduard K. de Jong, entitled "Non-Linear Execution of Application Program Instructions for Application Program Obfuscation", ~~Attorney Docket No. SUN-040025,~~ commonly assigned herewith.

Please replace paragraph **[0004]** with the following rewritten paragraph **[0004]**:

[0004] U.S. Patent Application Serial No. ~~_____~~10/673,021, filed September 25, 2003 in the name of inventor Eduard K. de Jong, entitled "Interleaved Data and Instruction Streams for Application Program Obfuscation", ~~Attorney Docket No. SUN-040026,~~ commonly assigned herewith.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Please replace paragraph **[0005]** with the following rewritten paragraph **[0005]**:

[0005] U.S. Patent Application Serial No. ~~———~~10/672,184, filed September 25, 2003 in the name of inventor Eduard K. de Jong, entitled "Rendering and Encryption Engine for Application Program Obfuscation", ~~Attorney Docket No. SUN-040027,~~ commonly assigned herewith.

Please replace paragraph **[0009]** with the following rewritten paragraph **[0009]**:

[0009] Figure 1 is a block diagram that illustrates a user device 120 comprising a processor configured to dispatch application program instructions based at least in part on an instruction set with a single opcode value encoding scheme. As shown in FIG. 1, a dispatcher 100 includes an instruction counter 125, an instruction executor 130, and an instruction fetcher 135. Instruction counter 125 maintains a reference to the next instruction to execute in an instruction stream 105 of an executable application program. As shown in FIG. 1, instruction stream 105 is represented as a table of (instruction number 155, opcode value 160) pairs, where the instruction number 155 is an index into the instruction stream 105, and the corresponding opcode value 160,140 is the opcode value stored at the location referenced by the instruction number 155. A single dispatch table 110 includes a reference 170 to the instruction implementation method 115 (the code that implements the instruction) ~~170~~ for each opcode value 165 of instructions in an instruction set. Instruction fetcher 135 receives an opcode value 175 from instruction counter 125 and uses the opcode value 175 to obtain a reference to the corresponding instruction implementation method (150) from dispatch table 110. Instruction fetcher 135 determines the instruction implementation method to execute (150) by performing a table lookup in the dispatch table 110 based at least in part on

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

the opcode value 145 of the instruction. Instruction executor 130 receives an instruction implementation method reference 150 from instruction fetcher 135 and executes the instruction implementation method. Unfortunately, the susceptibility of executable application programs 105 stored on unsecured devices means that an attacker knowing the instruction mapping used by the dispatch table 110 may obtain the executable application program 105. The executable application program 105 may then be executed on a user device controlled by the attacker, thus enabling unauthorized access to or use of a service.

Please replace paragraph **[0035]** with the following rewritten paragraph **[0035]**:

[0035] In operation, user device 305 issues an enrollment request 365 that includes a target ID. The target ID specifies a user device that will execute an obfuscated application program. The target ID may specify the user device 305 that issued ~~the~~ enrollment request 365. Alternatively, the target ID may specify a user device other than ~~the~~ user device 305 that issued ~~the~~ enrollment request 365. Application program provider 315 receives ~~the~~ enrollment request 365 and authenticator 355 authenticates user 300. If ~~the~~ user 300 is authenticated, authenticator 355 associates a secret 345 with the target ID and sends ~~the~~ secret 345, 370 to user device 305, which is saved as secret 325. According to one embodiment of the present invention, ~~the~~ application program provider 315 and ~~the~~ user device 305 determine at enrollment which obfuscation method to apply for each application program requested subsequently. According to another embodiment of the present invention, obfuscation methods are not determined during enrollment. Rather, in response to an application program request 375 comprising a target ID, application program provider 315 sends an obfuscated package 380 including an obfuscated

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

application program and obfuscation descriptor 385 to ~~the~~ user device 305 corresponding to the target ID.

Please replace paragraph **[0036]** with the following rewritten paragraph **[0036]**:

[0036] Still referring to FIG. 3, an enrolled user device 305~~300~~ obtains an obfuscated application program to execute by issuing an application program request 375 that includes the target ID 390. Application program provider 315 receives the application program request 375, determines which obfuscation method to apply based at least in part on the application program request 375, obtains the requested application program from application program database 350, and applies the obfuscation method to the application program. If the obfuscation methods to apply for each application program requested are determined during enrollment, application program provider 315 sends an obfuscated package 380 including an obfuscated application program to the user device 305 corresponding to the target ID 390. If the obfuscation methods are not determined during enrollment, application program provider 315 sends an obfuscated package 380 including an obfuscated application program and obfuscation descriptor 385 to the user device 305 corresponding to the target ID 390. The obfuscation descriptor 385 may indicate which obfuscation method was applied to the obfuscated application program 380. According to one embodiment of the present invention, secret 345 is used to encrypt the obfuscation descriptor 385. The obfuscation descriptor 385 may be sent separately from the obfuscated package 380. Alternatively, the obfuscation descriptor 385 may be embedded within the obfuscated application program.

Please replace paragraph **[0039]** with the following rewritten paragraph **[0039]**:

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

[0039] User device 305 may be any device configured to render digital content to a user 305 using an obfuscated application program. The digital content may be rendered, by way of example, by visual, auditory, or tactile means. Exemplary user devices include one or more of the following devices configured to render digital content to a user 305 using an obfuscated application program: a personal digital assistant (PDA) 330, a personal computer (PC) 335, a mobile phone 340, a server computer in communication with a user display, or the like.

Please replace paragraph **[0040]** with the following rewritten paragraph **[0040]**:

[0040] Figure 4 is a block diagram that illustrates a system for application program obfuscation using a secure portable device in accordance with one embodiment of the present invention. As shown in FIG. 4, user device 405 comprises a secure portable device such as a Java Card™ technology-enabled device, or the like. The system illustrated by FIG. 4 includes at least one user device 405 in communication with at least one application program provider 415 via network 410. User device 405 includes a virtual machine 485 and a smart card 490. Smart card 490 also comprises a secret 425 established upon successful enrollment with application program provider 415. Deobfuscator 420 comprises a first portion 482 in virtual machine 485 and a second portion 484 in smart card 425. First portion 484 uses secret 425 to create permuted dispatch tables. ~~First~~Second portion 482 uses the permuted dispatch tables to execute the obfuscated code.

Please replace paragraph **[0043]** with the following rewritten paragraph **[0043]**:

[0043] In operation, user device 405, which can be, for example, any one of a personal digital assistant (PDA) 430,

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

a personal computer (PC) 435, a mobile phone 440, issues an enrollment request 465 that includes a virtual machine identifier (VM ID) ~~495~~. The VM ID ~~495~~ specifies a user device that will execute an obfuscated application program. The VM ID ~~495~~ may be based at least on a target ID, and secret 425 may be based at least in part on VM ID ~~495~~. The VM ID may specify the VM ID of the user device 405 that issued the enrollment request 465. Alternatively, the VM ID may specify a user device other than the user device 405 that issued the enrollment request 465. Application program provider 415 receives the enrollment request 465 and authenticator 455 authenticates user 400. If the user 400 is authenticated, authenticator associates a secret 445 with the VM ID ~~495~~ and sends the secret 445, 470 to user device 405. According to one embodiment of the present invention, the application program provider 415 and the user device 405 determine at enrollment which obfuscation method to apply for each application program requested subsequently. According to another embodiment of the present invention, obfuscation methods are not determined during enrollment. Rather, in response to an application program request 475 comprising a VM ID, application program provider 415 sends an obfuscated package 480 including an obfuscated application program and obfuscation descriptor 485 to the user device 405 corresponding to the target ID.

Please replace paragraph **[0051]** with the following rewritten paragraph **[0051]**:

[0051] Figures 5A and 5B are block diagrams that illustrate obfuscated package data structures in accordance with embodiments of the present invention. Figure 5A illustrates an obfuscated package data structure 500 having an obfuscated application program 505. Figure 5B illustrates an obfuscated package data structure 510~~500~~ having an obfuscated application program 525, an obfuscation descriptor 530, protected data 520 and a cryptographic key

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

515. Obfuscation descriptor 530 includes information about the obfuscation method used to create obfuscated application program 525. A cryptographic process is applied to protected data 520 together with cryptographic key 515.

Please replace paragraph **[0052]** with the following rewritten paragraph **[0052]**:

[0052] Figure 5C is a block diagram that illustrates creating an obfuscated package in accordance with one embodiment of the present invention. An obfuscator 555 of an application program provider receives application program information comprising application program code 535 and application program data 540. The application program data 540 may comprise data referenced during execution of application program code 535. Application program data may also comprise data used to initialize the application program. Obfuscator 555 then applies an obfuscation method, identified by obfuscation descriptor 550, to the application program code 535, the application program data 540, or both, to create an obfuscated application program 565. The obfuscated application program 565 and possibly other data 545 together form an obfuscated package 560.

Please replace paragraph **[0085]** with the following rewritten paragraph **[0085]**:

[0085] Turning now to FIG. 21A, a block diagram that illustrates a linear application program execution order is presented. As shown in FIG. 21A, the location of the next instruction to execute can be determined based at least in part on the current instruction. If the current instruction is a jump instruction (2155, 2160), the next instruction is the address specified in the jump instruction (2165, 2170). If the current instruction is not a jump instruction, the next instruction to execute is determined by advancing~~advanceing~~ the current instruction address.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Please replace paragraph **[0086]** with the following rewritten paragraph **[0086]**:

[0086] Turning now to FIG. 21B, a detailed block diagram that illustrates application program obfuscation by nonlinear application program execution order in accordance with one embodiment of the present invention is presented. User device 2135 comprises an instruction permuter 2110 that receives an instruction counter value 2195 and permutes it to create a permuted instruction counter value 2182. The permuted instruction counter value 2182 is used to access the address of the instruction to execute in instruction stream memory 2115~~2155~~. A dispatch table 2185 maintains an association between opcode values and references to instruction implementation methods 2190. According to one embodiment of the present invention, the instruction permutation may be based at least in part on the size of an instruction cache.

Please replace paragraph **[0094]** with the following rewritten paragraph **[0094]**:

[0094] Referring now to FIG. 25, a permutation set 2545 comprises N low order bits 2515 and M high order bits 2540 of current instruction counter 2500. The low order bits 2515 are used as an index into an instruction location permutation table 2505. Instruction location permutation table 2505 comprises indexed entries that indicate a modifier value 2520 to use for the high order bits 2530 of the permuted instruction counter 2510. Permuted instruction counter 2510 is initialized to the same value as current instruction counter 2500. The M high order bits 2540 in the current instruction counter 2540 are shifted or moved to the M low order bits of permuted instruction counter 2510. As shown in FIG. 25, the value of the N low order bits "001" (2525~~2515~~) is associated with modifier value "100" (2520).

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Thus, the N high order bits 2530 of permuted instruction counter 2510 are replaced with modifier value "100" (2520).

Please replace paragraph **[0096]** with the following rewritten paragraph **[0096]**:

[0096] Referring now to FIG. 27, a permutation set 2745 comprises N high order bits 2715 and M low order bits 2740 of current instruction counter 2700. The high order bits 2715 are used as an index into an instruction location permutation table 2705. Instruction location permutation table 2705 comprises indexed entries that indicate a modifier value 2720 to use for the low order bits 2730 of the permuted instruction counter 2710. Permuted instruction counter 2710 is initialized to the same value as current instruction counter 2700. The M low order bits 2740 in the current instruction counter 2740 are shifted or moved to the M high order bits of permuted instruction counter 2710. As shown in FIG. 27, the value of the N high order bits "011" (2725~~2735~~) is associated with modifier value "101" (2720). Thus, the N low order bits 2730 of permuted instruction counter 2710 are replaced with modifier value "101" (2720).

Please replace paragraph **[0098]** with the following rewritten paragraph **[0098]**:

[0098] Figure 29 illustrates using multiple instruction location permutation tables to create a permuted instruction counter. As shown in FIG. 29, a first permutation set 2960 comprises N_1 low order bits 2915 and M_1 high order bits 2940 of current instruction counter 2900. The low order bits 2915 are used as an index into a first instruction location permutation table 2905. Instruction location permutation table 2905 comprises indexed entries that indicate a modifier value 2920 to use for the high order bits 2930 of the permuted instruction counter 2910. Permuted instruction counter 2910 is initialized to the same value as current

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

instruction counter 2900. The M_1 high order bits 2940 in the current instruction counter 2940 are shifted or moved to the M_1 low order bits of permuted instruction counter 2910. As shown in FIG. 29, the value of the N_1 low order bits "001" (2925~~2915~~) is associated with modifier value "100" (2920). Thus, the N_1 high order bits 2930 of permuted instruction counter 2910 are replaced with modifier value "100" (2920).

Please replace paragraph **[0101]** with the following rewritten paragraph **[0101]**:

[0101] Figures 23 - 29 are for purposes of illustration only and are not intended to be limiting in any way. Those of ordinary skill in the art will recognize that the number of bits used to index an instruction location permutation table 2305 (Fig. 23)~~2105~~ may be more than or less than what is illustrated. Furthermore, the particular bits selected to index the instruction location permutation table 2305~~2105~~, as well as the particular bits selected to be modified may differ from what is shown in FIGS. 23 - 29. Additionally, more than two instruction location permutation tables may be used.

Please replace paragraph **[0106]** with the following rewritten paragraph **[0106]**:

[0106] Turning now to FIG. 31, a detailed block diagram that illustrates application program obfuscation by nonlinear execution of an application program having application program instructions interleaved with application program data in an instruction stream in accordance with one embodiment of the present invention is presented. User device 3145 comprises an instruction location permutation table 3105 that includes multiple entries, where each entry determines how instruction counter 3120 is modified to create a modified instruction counter. The modified instruction counter is used to access an instruction to execute. Similarly, data location

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

permutation table ~~3115~~3110 includes multiple entries, where each entry determines how data location counter 3125 is modified to create a modified data location counter. The modified data location counter is used to access the data referenced by the instruction to execute.

Please replace paragraph **[0108]** with the following rewritten paragraph **[0108]**:

[0108] According to one embodiment of the present invention, the application program data interleaved into instruction stream memory 3115 comprises at least one cryptographic key for use in decrypting protected data. The protected data may be stored elsewhere within the interleaved instruction stream memory ~~3114~~3115. The protected data may also be stored in another memory on the user device.

Please replace paragraph **[0113]** with the following rewritten paragraph **[0113]**:

[0113] At 3230, program code 3200 and serialized program data elements from process 3215 are optionally encoded with one or more opcode permutations. If data elements are interleaved in the instruction stream, the program code received at 3230 is first modified at 3220 so the data access instructions reference data elements in the instruction stream. The particular encoding method used at 3230 is determined by an obfuscation descriptor 3225. The obfuscation descriptor 3225 also determines an instruction stack address 3240 and a data stack address ~~3260~~3250. At 3235, the serialized program data elements are padded with one or more randomized bytes 3210 and encoded as pseudo opcodes. At 3245, the encoded code from 3230 and the encoded data from 3235 are appended into a single instruction stream. At 3255, an instruction location permutation is applied to one or more instruction locations to create an obfuscated package or program 3260.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888

Please replace paragraph **[0125]** with the following rewritten paragraph **[0125]**:

[0125] Turning now to FIG. 38, a flow diagram that illustrates a method for application program obfuscation from the perspective of an application program provider in accordance with one embodiment of the present invention is presented. At 3800, a reference to a decryption algorithm and a first cryptographic content key are received. The decryption algorithm may be any decryption algorithm known in the art. At 3805, a key decryption program that performs the decryption algorithm for the first cryptographic content key is created. At 3810, a cryptographic process is applied to a second cryptographic content key together with the first cryptographic content key to create an encrypted second cryptographic content key. At 3815, the encrypted second cryptographic content key is scrambled into the instruction stream using a code obfuscation method to create an obfuscated key decryption program. The code obfuscation method may be indicated by an obfuscation descriptor. The second cryptographic content key may be scrambled into the instruction stream as was described above with reference to FIGS. 31-35. According to one embodiment of the present invention, a cryptographic process is applied to the obfuscated application program together with a cryptographic key, to create an encrypted obfuscated application program. At 3820~~3825~~, the obfuscated key decryption program having the encrypted second cryptographic content key scrambled in the instruction is sent to a target device. The obfuscated key decryption program may be sent together with digital content protected by the second cryptographic content key. Alternatively, the obfuscated key decryption program and the protected digital content may be sent separately.

GUNNISON, McKAY & HODGSON, L.L.P.
Garden West Office Plaza
1900 Garden Road, Suite 220
Monterey, CA 93940
(831) 655-0880
Fax (831) 655-0888